White Paper

DOCUMENTUM CONTENT SERVER CENTRAL AUTHENTICATION SERVICE (CAS) SSO

A Detailed Review

Abstract

This white paper describes Central Authentication Service (CAS) based Content Server environment, CAS authentication process and explains CAS server and Content Server configuration. It includes detailed steps to deploy the system and troubleshoot issues.

November, 2013

EMC²

Copyright © 2013 EMC Corporation. All Rights Reserved.

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

The information in this publication is provided “as is.” EMC Corporation makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose.

Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

For the most up-to-date listing of EMC product names, see EMC Corporation Trademarks on EMC.com.

Part Number h12009

Table of Contents

Executive Summary 4
Audience 4
Terminology 4
CAS Architecture 4
Authentication Process 5
CAS Server Configuration & Deployment 10
Tomcat Setup for deploying CAS 10
Install Tomcat 10
Enable SSL for Tomcat 10
Store Public Certificates in Trust-Store 11
Deploy CAS Application 13
Configure CAS Application 14
Modify WEB-INF/cas.properties 14
Configure LDAP for CAS 14
Customize CAS Response 15
Service Management 17
Content Server Configuration 20
CAS Plug-in Binary 20
CAS Plug-in Trust-Store 20
CAS Plug-in Configuration 20
Verify Plug-in load 21
LDAP Configuration 22
Troubleshooting 22
CAS Plug-in Trace 22
CAS Server Logs 22
SSL related Errors 22
Error when trying to access Service Management webapp 22
Error during CAS server's callback to Application Server 23
Error during login to CAS Server 23
CAS Server Errors 23
CAS Server responds with Invalid Ticket message 23
CAS Plug-in Errors 24
Connection to CAS Server fails in non-anonymous SSL mode 24
Plug-in load fails 25
Conclusion 26
References 26

Executive Summary

Central Authentication Service (CAS) is an enterprise level, open-source, single sign-on (SSO) solution. CAS is http based protocol that requires each of its components to be accessed through specific URIs. CAS was originally developed by Yale University for Single Sign on. Documentum Content Server is providing support for CAS SSO by introducing authentication plug-in for CAS. Content Server authentication supports CAS protocol 2.0. This paper explains CAS Architecture, CAS Protocol and configuration required at CAS Server and Content Server for enabling CAS SSO.

Audience

This white paper is intended for engineers, support professionals and customers. It provides detailed understanding for enabling CAS SSO for Documentum applications.

Terminology

Special terms, abbreviations and acronyms that may appear in this guide are defined below:

Term	Description
Central Authentication Service (CAS)	Open source single sign-on for web and cloud based solutions
Single sign-on (SSO)	Authentication process that allows user to provide his credentials once in order to access multiple applications
Service	Service is an application that accepts CAS authentication
Ticket Granting Ticket (TGT)	Ticket indicating user has successfully logged-in to CAS server
Service Ticket (ST)	Ticket sent by CAS to a service for identifying that service
Proxy Granting Ticket (PGT)	Ticket sent by CAS to service with valid ST for requesting Proxy Tickets
Proxy Ticket (PT)	Ticket used by proxy service to access target service for multi-tier authentication
Certificate Authority (CA)	Certificate Authority is an entity that issues digital certificates
Keystore	File containing public certificate and private key
Trust-Store	File containing public certificates from trusted servers

CAS Architecture

CAS protocol involves four parties as shown in Figure 1: client web browser, the web application requesting authentication (Service / Proxy Service), back-end application (Target Service) from which web application needs some data and CAS server. Here, Content Server is the back-end service or target service that web application is trying to access by requesting Content Server session. Web application is acting as a proxy

and uses CAS proxy ticket to request Content Server session. For user authentication, application redirects the user to CAS server against which user authenticates by providing its credentials. After successful authentication, application requests for proxy ticket from CAS Server. Content server authentication plug-in for CAS eliminates the need for user to authenticate again with Content Server by accepting CAS proxy ticket, thereby enabling SSO. Content Server verifies proxy ticket sent by application with CAS server and returns session on successful authentication.

CAS server supports multiple Identity Providers like LDAP/ Active Directory, Kerberos, RDBMS etc. and delegates authentication decisions to these servers.

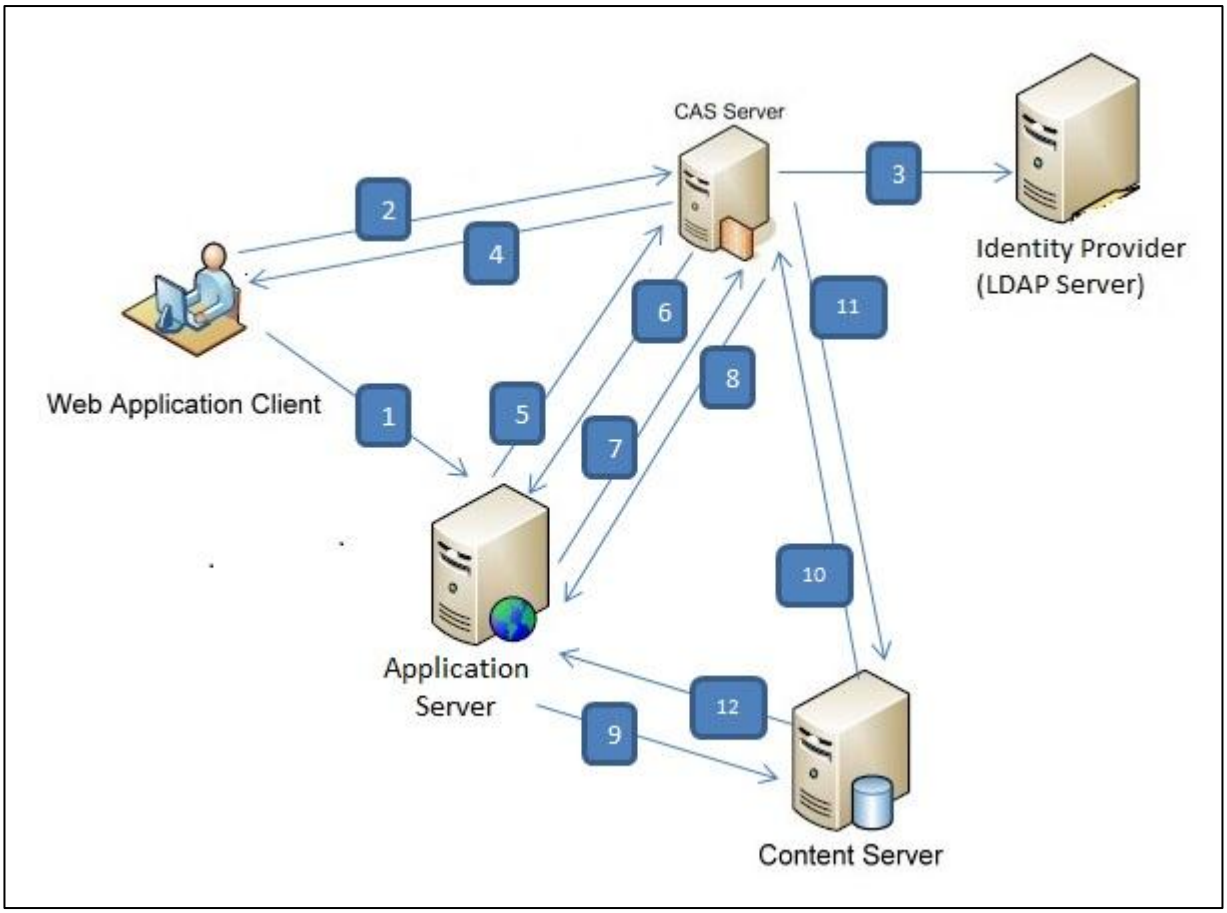

Figure 1. System Deployment and CAS Authentication Flow

Authentication Process

The authentication process shown in Figure 1 is as follows:

1. User accesses client application deployed on application server and tries to access protected resource for which application needs Content Server session. If

user has already authenticated with CAS server and has Ticket Granting Ticket (TGT) cookie set, then control moves to step 4.

2. Application asks user to authenticate with CAS Server and user is redirected to CAS login page (/login). User enters his username and password. User credentials along with service URL are sent to CAS server. Service URL is URL of resource on application that user is trying to access. Login request looks like this:

   ```
   <cas_server_url>/login?service=<service_url>
   ```

3. CAS Server verifies whether application is allowed to use CAS by checking that service is registered in CAS Service Registry. The Service Registry component is responsible for defining the allowed services that may request and validate tickets provided by CAS. If matching entry is found in Service Registry, CAS Server verifies user credentials against Identity Provider.

4. On successful authentication, CAS Server returns Service Ticket by redirecting to application (service URL) with Service Ticket. Also, CAS server sets TGT cookie on user session to indicate that user has successfully logged in and would not be asked to login for subsequent requests (till session expires).

5. Application then calls CAS server requesting Proxy Granting Ticket (PGT) by sending Service Ticket, application callback URL (pgtUrl) & service URL. Service URL should be same as that sent while requesting Service Ticket. Callback URL is used by CAS server to verify service (application) identity and return Proxy Ticket. Callback URL must be HTTPS since CAS verifies that both that the SSL certificate is valid and that its name matches that of the service to verify service identity. Application makes a request like below:

   ```
   <cas_server_url>/serviceValidate?service=<service_url>&ticket=<servic
   e_ticket>&pgtUrl=<callback_url>
   ```

6. CAS server verifies that Service Ticket is valid and was generated for the same Service URL. CAS server then invokes callback URL and sends Proxy Granting Ticket as request parameter (PGTID).

   CAS server responds to application with PGTIOU (I owe you PGT). CAS server invokes callback URL and passes PGTID & same PGTIOU. Application then retrieves PGTID by comparing PGTIOU.

   CAS server response looks like below:

   ```
   <cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>
      <cas:authenticationSuccess>
         <cas:user>username</cas:user>
         <cas:proxyGrantingTicket>PGTIOU-4-YO2XOuDgmMiD3Ewllaew
         </cas:proxyGrantingTicket>
     </cas:authenticationSuccess>
   </cas:serviceResponse>
   ```

CAS server GET request to callback URL looks like below:

```
<callback_url>/pgtCallback?pgtIou=<PGTIOU>&pgtId=<PGTID>
```

7. Application then requests Proxy Ticket from CAS server by sending Proxy Granting Ticket and Target Service Name. Target Service Name should be same as that configured for CAS authentication plug-in. CAS plug-in configuration is explained later on page 20. Target Service should be registered in CAS Service Registry.

   Application request looks like below:

   ```
   <cas_server_url>/proxy?pgt=<PGT>&targetService=ContentServer
   ```

8. CAS server verifies that Target Service is registered in service registry. CAS server then verifies Proxy Granting Ticket and if Proxy Granting Ticket is still valid, returns Proxy Ticket to application.

   CAS server response looks like below:

   ```
   <cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>
      <cas:proxySuccess>
         <cas:proxyTicket>ST-10-d4dYIOtvDzNqzJO9Ml9p</cas:proxyTicket>
      </cas:proxySuccess>
   </cas:serviceResponse>
   ```

9. Application then uses the same username which was used to authenticate with CAS server and proxy ticket sent by CAS Server to request session from Content Server. Password should be in below format:

   ```
   Password Format: DM_PLUGIN=dm_cas/<proxy_ticket>
   ```

10. Session request is redirected to CAS authentication plug-in that validates proxy ticket with CAS Server. CAS authentication plug-in sends Target Service name configured in configuration file along with Proxy Ticket to CAS Server for validating Proxy Ticket.

    Plug-in request to CAS server looks like below:

    ```
    <cas_server_url>/proxyValidate?service=<service_name>&ticket=<proxy_t
    icket>
    ```

11. CAS server validates that Proxy Ticket is valid and was generated for the same service and returns user distinguished name, an LDAP property of CAS user with authentication success response.

    CAS sever response looks like below:

    ```
    <cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>
       <cas:authenticationSuccess>
          <cas:user>user</cas:user>
    ```

```
        <cas:attribute name="dmCSLdapUserDN"
        value="CN=user,OU=testou,DC=iigplat,DC=com"/>
        <cas:proxies>
           <cas:proxy>[callback_url]</cas:proxy>
        </cas:proxies>
     </cas:authenticationSuccess>
  </cas:serviceResponse>
```

12. CAS Plug-in compares user distinguished name (DN) with user DN stored in Docbase for that user. If match is successful, then Content Server session is returned to the application. In typical deployment scenario, Content Server uses the same Identity Provider (e.g. LDAP) to sync its user as that used by CAS Server, so user DN will be same as that that sent by CAS Server.

Below flow chart explains the CAS protocol. Steps in blue box are activities performed by CAS whereas steps in green box are user / application triggers.

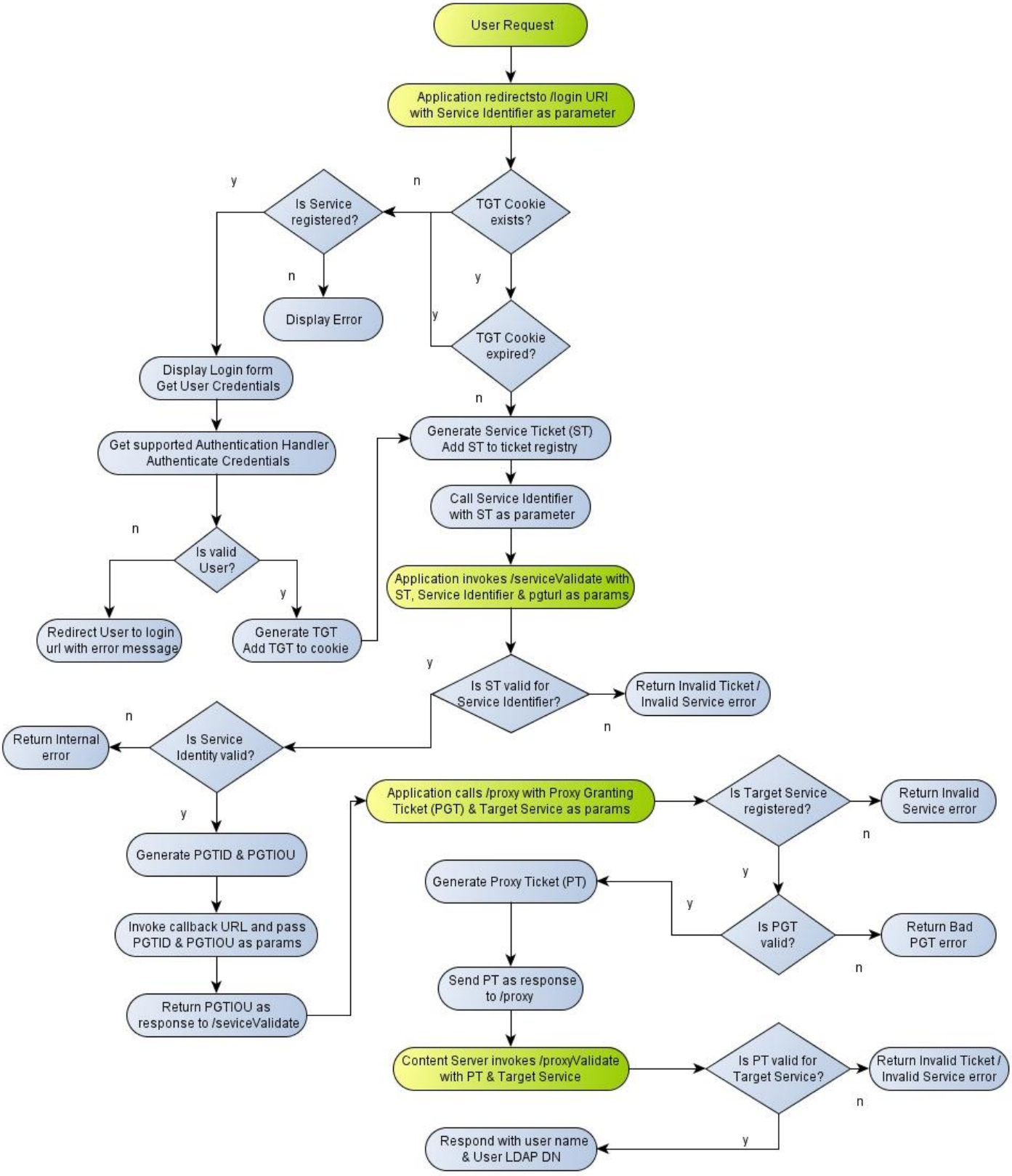
User Request
Application redirectsto /login URI with Service Identifier as parameter
TGT Cookie exists?
Is Service registered?
Display Error
TGT Cookie expired?
Display Login form Get User Credentials
Get supported Authentication Handler Authenticate Credentials
Is valid User?
Redirect User to login url with error message
Generate TGT Add TGT to cookie
Generate Service Ticket (ST) Add ST to ticket registry
Call Service Identifier with ST as parameter
Application invokes /serviceValidate with ST, Service Identifier & pgturl as params
Is ST valid for Service Identifier?
Return Invalid Ticket / Invalid Service error
Is Service Identity valid?
Return Internal error
Generate PGTID & PGTIOU
Invoke callback URL and pass PGTID & PGTIOU as params
Return PGTIOU as response to /seviceValidate
Application calls /proxy with Proxy Granting Ticket (PGT) & Target Service as params
Is Target Service registered?
Return Invalid Service error
Generate Proxy Ticket (PT)
Is PGT valid?
Return Bad PGT error
Send PT as response to /proxy
Content Server invokes /proxyValidate with PT & Target Service
Is PT valid for Target Service?
Return Invalid Ticket / Invalid Service error
Respond with user name & User LDAP DN
y
n

Figure 2. CAS Protocol 2.0 Flow Chart

CAS Server Configuration & Deployment

This section describes steps to setup, configure and deploy CAS Server. CAS is available as web application that needs to be deployed on Application Server. This paper explains CAS deployment steps using Apache Tomcat as Application Server. This paper shows the minimal configuration required for CAS Server to work as authentication service for Content Server.

Tomcat Setup for deploying CAS

Install Tomcat

Download and install Apache Tomcat Server. Select proper location for Java, CAS 3.5 requires JDK version 1.6 or higher.

Enable SSL for Tomcat

Create SSL certificate for CAS Server

For purpose of this paper, below steps show how to create self-signed SSL certificate.

```
keytool -genkey -alias cas -keyalg RSA -keystore c:\ks\caskeystore
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  casserver
What is the name of your organizational unit?
  [Unknown]:  iig
What is the name of your organization?
  [Unknown]:  emc
What is the name of your City or Locality?
  [Unknown]:  blr
What is the name of your State or Province?
  [Unknown]:  kn
What is the two-letter country code for this unit?
  [Unknown]:  in
Is CN=casserver, OU=iig, O=emc, L=blr, ST=kn, C=in correct?
  [no]:  yes

Enter key password for <cas>
        (RETURN if same as keystore password):
```

It is important to set the value of the "first and last name" field above to fully qualified domain name of the machine.

Enable SSL port

Edit server.xml inside configuration directory and uncomment SSL Connector. Add location to keystore created in previous step and keystore password.

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
           maxThreads="150" scheme="https" secure="true"
           clientAuth="false" sslProtocol="TLS"
           keystoreFile="C:/ks/caskeystore" keystorePass="changeit"/>
```

When using JSSE configuration, APR library loader needs to be commented:

```
<Listener className="org.apache.catalina.core.AprLifecycleListener"
SSLEngine="on" />
```

Store Public Certificates in Trust-Store

CAS Server, Application server & LDAP server's public certificates need to be imported to CAS Server's trust-store. These certificates can also be imported to trust-store of JRE used by CAS Server, for same behavior. Steps to import public certificates to JRE trust-store are explained below.

CAS Server's SSL Certificate

CAS Server's SSL Certificate needs to be in trust-store to access Service Manager webapp, else SSLHandshakeException is thrown by CAS server.

Export CAS Server public certificate from CAS server's keystore as shown:

```
keytool -export -keystore caskeystore -alias cas -file cas.cer
Enter keystore password:
Certificate stored in file <cas.cer>
```

Import server's public certificate & certificate chain to local keystore (trust-store of JRE used by application server).

```
keytool -import -trustcacerts -alias cas -keystore
"%JAVA_HOME%\jre\lib\security\cacerts" -file cas.cer
```

```
Enter keystore password:

Owner: CN=cs7179, OU=iig, O=emc, L=blr, ST=kn, C=in

Issuer: CN=cs7179, OU=iig, O=emc, L=blr, ST=kn, C=in

Serial number: 2af72204

Valid from: Thu Jun 13 23:46:16 PDT 2013 until: Wed Sep 11 23:46:16 PDT
2013

Certificate fingerprints:

MD5:  25:9B:07:7D:AD:7B:8F:F6:A8:13:15:F9:7B:9C:0E:62

SHA1: BB:FA:8E:0B:B5:96:7C:6B:C1:87:7C:55:E0:8F:0D:99:72:10:D7:25

SHA256:
9F:A0:31:DB:BE:00:80:80:47:A1:55:44:73:FC:76:0E:AC:BF:D4:2A:FE:01:8B:6A:
77:1C:F1:A4:6C:3A:B1:1D

Signature algorithm name: SHA256withRSA

Version: 3

Trust this certificate? [no]:  yes

Certificate was added to keystore
```

Application Server's SSL Certificate

CAS server verifies identity of server requesting proxy granting ticket by verifying its public certificate. So, application server's SSL certificate must be placed in trust-store.

Import Application server's public certificate to JRE's trust-store as shown:

```
keytool -import -trustcacerts -alias appserver -keystore
"%JAVA_HOME%\jre\lib\security\cacerts" -file cascrt.der

Enter keystore password:

Owner: EMAILADDRESS=cas@dctm.com, CN=cs7163.iigplat.com, OU=IIG, O=EMC,
ST=Kar, C=IN

Issuer: EMAILADDRESS=ica1@dctm.com, CN=ICA1, OU=IIG, O=EMC, ST=Kar, C=IN

Serial number: 10

Valid from: Thu Jun 06 04:23:26 PDT 2013 until: Fri Jun 06 04:23:26 PDT
2014

Certificate fingerprints:

MD5:  AD:34:2C:25:B8:12:9D:A1:DD:18:31:1B:D4:42:30:CE

SHA1: 19:C5:AD:77:41:A7:11:71:C8:4A:DF:8A:14:B8:17:6E:C2:62:74:12

SHA256:
EA:47:A6:C1:FA:A9:2B:C7:58:73:81:F8:0A:3C:BB:20:61:DC:0B:04:6B:4C:03:AA:
16:37:C6:1F:D3:FA:AB:FC

Signature algorithm name: SHA1withRSA

Version: 3
```

```
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

LDAP Server's SSL Certificate

CAS server needs to trust the LDAP server acting as Identity Provider for secure connection to LDAP Server over SSL. Import LDAP SSL Certificate to trust-store as shown:

```
keytool -import -trustcacerts -alias ldap -keystore
"%JAVA_HOME%\jre\lib\security\cacerts" -file
domainctlr.iigplat.com_iigplat-DOMAINCTLR-CA.crt
Enter keystore password:
Owner: CN=iigplat-DOMAINCTLR-CA, DC=iigplat, DC=com
Issuer: CN=iigplat-DOMAINCTLR-CA, DC=iigplat, DC=com
Serial number: 7d98e408fdb6b3814570ec0776bdada7
Valid from: Mon Jun 17 07:07:34 PDT 2013 until: Sun Jun 17 07:17:33 PDT
2018
Certificate fingerprints:
MD5:  E7:53:8B:96:60:A2:8D:89:A1:F2:38:FC:9C:A0:6E:82
SHA1: E5:69:CA:5F:81:35:46:76:6C:2F:30:A2:DA:DA:BD:3F:46:1C:65:9B
SHA256:
B5:09:B7:38:62:71:5F:DE:C2:20:D5:D9:5B:27:FD:93:CA:C4:65:7B:7E:7F:41:9B:
FA:3F:E5:9D:90:F3:C8:13
Signature algorithm name: SHA1withRSA
Version: 3
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

Deploy CAS Application

While building CAS, make sure LDAP module is enabled. For creating this setup, cas release build was used in which all modules are enabled by default. Documentum Content Server release 7.1 supports CAS Server version 3.5.2.

- Download CAS server 3.5.2 from http://www.jasig.org/cas/download
- Create a folder 'cas' in Application Server deployment directory (webapps).
- Extract CAS server 3.5.2 archive file (cas-server-3.5.2-release).
- Extract the content of cas-server-3.5.2\modules\cas-server-uber-webapp-3.5.2.war to 'cas' directory

Configure CAS Application

This section defines the minimum configuration required for CAS to work as authentication provider for Content Server with LDAP as Identity Provider.

Modify WEB-INF/cas.properties

Update below properties:

server.name	URL that is used to access CAS Server. It is recommended to use https in deployment environments (e.g. https://cs7163.iigplat.com:8443)
host.name	Full hostname of CAS Server host (e.g. cs7163.iigplat.com)
server.prefix	${server.name}/‹application_name› Replace ‹application_name› with CAS application name (e.g. ${server.name}/cas)

Configure LDAP for CAS

All LDAP related configuration needs to be done in WEB-INF/deployerConfigContext.xml

Define ContextSource bean for LDAP

This bean is used for connecting to LDAP for performing authentication operations. LDAP Server url, username and password needs to be configured for CAS Server connections to LDAP as given in below example. If LDAP is configured over SSL, then LDAP url starts with *ldaps* (e.g. ldaps://‹hostname›:636) else LDAP url starts with *ldap* (e.g. ldap://‹hostname›:389).

```
<bean id="contextSource"
      class="org.springframework.ldap.core.support.LdapContextSource">
   <property name="pooled" value="false"/>
   <property name="url" value="ldaps://domainctlr.iigplat.com:636" />
   <property name="userDn"
             value="CN=Administrator,CN=Users,DC=iigplat,DC=com"/>
   <property name="password" value="password"/>
   <property name="baseEnvironmentProperties">
      <map>
         <entry key="com.sun.jndi.ldap.connect.timeout" value="3000" />
         <entry key="com.sun.jndi.ldap.read.timeout" value="3000" />
         <entry key="java.naming.security.authentication"
```

```
                value="simple" />
        </map>
    </property>
</bean>
```

Add authentication handler for LDAP

Authentication handler needs to be defined for every Identity Provider. Authentication handler for LDAP (Microsoft Active Directory) is defined as below. Put this bean inside *authenticationHandlers* property of *authenticationManager* bean.

```
<bean class="org.jasig.cas.adaptors.ldap.BindLdapAuthenticationHandler"
      p:filter="sAMAccountName=%u"
      p:searchBase="OU=testou,DC=iigplat,DC=com"
      p:contextSource-ref="contextSource"
      p:ignorePartialResultException="true" />
```

For more details on LDAP configuration, refer LDAP section in CAS user manual (https://wiki.jasig.org/display/CASUM/LDAP).

Customize CAS Response

By default, on successful authentication for proxy ticket validation request, CAS server responds with just user name in the response message. CAS response needs to be customized such that it includes full user DN (as shown in step 11 of authentication process), as it's needed by Content Server for user verification.

Add CredentialsToPrincipalResolvers bean for LDAP

A principal describes an authenticated user. Principal contains attributes describing the user. CredentialsToPrincipalResolver component helps to map credential attributes onto a Principal. Principal is used by view to create response with user attributes defined in *AttributeRepository* bean. Put below bean inside *credentialsToPrincipalResolvers* property of *authenticationManager* bean.

```
<bean
class="org.jasig.cas.authentication.principal.CredentialsToLDAPAttribute
PrincipalResolver">
    <property name="credentialsToPrincipalResolver">
        <bean
        class="org.jasig.cas.authentication.principal.UsernamePasswordCred
        entialsToPrincipalResolver"/>
```

```
    </property>
    <property name="filter" value="(sAMAccountName=%u)"/>
    <property name="principalAttributeName" value="sAMAccountName"/>
    <property name="searchBase" value="OU=testou,DC=iigplat,DC=com"/>
    <property name="contextSource" ref="contextSource"/>
   <property name="attributeRepository" ref="attributeRepository"/>
</bean>
```

Replace AttributeRepository Bean

Replace *AttributeRepository* bean stub with *AttributeRepository bean* defined in below example, which defines the attributes that CAS returns to Content Server. Attribute 'dmCSLdapUserDN' needs to be added in *resultAttributeMapping* property, whose value will be set to user distinguished name and is matched against user dn value stored in docbase.

```
<bean id="attributeRepository"
class="org.jasig.services.persondir.support.ldap.LdapPersonAttributeDao"
>
    <property name="contextSource" ref="contextSource" />
    <property name="baseDN" value="OU=testou,DC=iigplat,DC=com" />
    <property name="requireAllQueryAttributes" value="true" />
    <property name="queryAttributeMapping">
       <map>
          <entry key="username" value="sAMAccountName" />
       </map>
    </property>
    <property name="resultAttributeMapping">
       <map>
          <entry value="dmCSLdapUserDN" key="distinguishedName"/>
       </map>
    </property>
</bean>
```

Update View

View needs to be updated such that it includes user distinguished name in the response sent to Content Server for proxy ticket validation request.

Add below lines after ‹cas:user› … ‹/cas:user› in file **WEB-INF/view/jsp/protocol/2.0/casServiceValidationSuccess.jsp**:

```
<c:forEach var="auth" items="${assertion.chainedAuthentications}">
   <c:forEach var="attr" items="${auth.principal.attributes}" >
      <cas:attribute name="${fn:escapeXml(attr.key)}"
                     value="${fn:escapeXml(attr.value)}"/>
   </c:forEach>
</c:forEach>
```

Service Management

While creating proxy tickets and validating proxy tickets, CAS server verifies whether target service for which the request is made is registered or not. This section shows how to register service for Content Server with CAS server.

Admin User

Create Admin User for Service Management application of CAS. Edit userDetailsService bean in WEB-INF/deployerConfigContext.xml and modify user name. This user should be valid user and CAS authentication should work for this user.

```
<sec:user-service id="userDetailsService">
   <sec:user name="admin" password="notused" authorities="ROLE_ADMIN" />
</sec:user-service>
```

Create Services

For creating services CAS provides service management webapp. To access it, first CAS server needs to be deployed.

1. Start CAS Server
2. Access ‹cas_server_url›/services.
3. It will ask for user credentials. Login using admin account created.
4. Page with list of services added to Service Registry is displayed.
5. Click on 'Add New Service' or access ‹cas_server_url›/services/add.html.

6. Fill the form to create new service and save service. Put name in 'Service URL' same as that configured in CAS plugin configuration file (as shown in Figure 3).

7. Verify that service is created

Tip: By default there is in-memory data store for service registry. This should be changed to some persistent store in deployment environment. Else, services added in above steps will be lost on server restart.

Tip: Service also needs to be registered for Application server (to access service management webapp). By default, service definition exists that allows http(s) & imap(s) protocols. In deployment environment, this must be modified too.

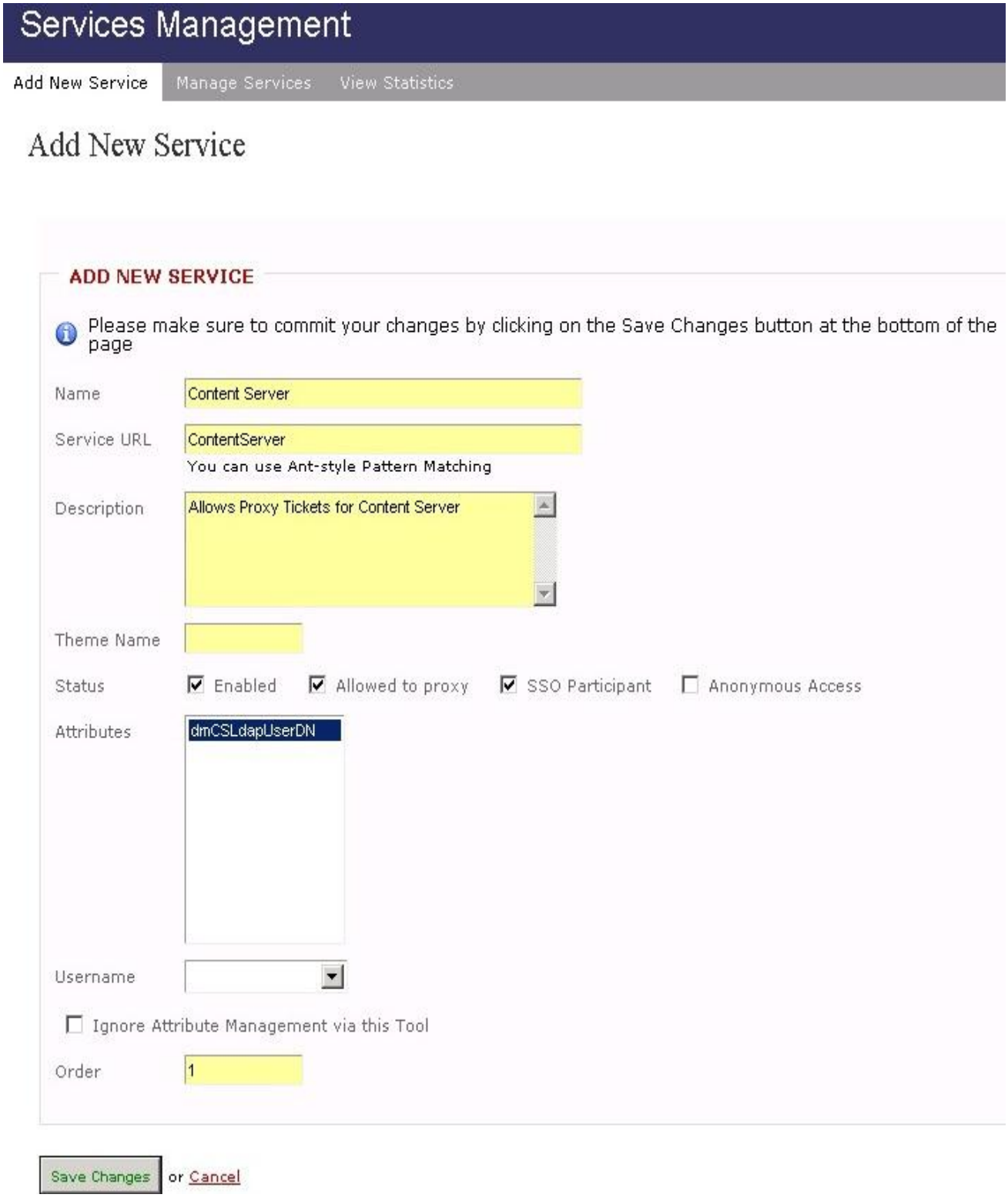

Figure 3. CAS Page to add new Service

Content Server Configuration

This section explains setup required at Content Server side for CAS authentication plug-in.

CAS Plug-in Binary

CAS plug-in needs to be copied to 'auth' directory. Plug-ins inside *auth* directory are automatically loaded on server startup.

1. Stop all Repository Services
2. Copy CAS Plugin binary (**dm_cas_auth.dll**) from **%DM_HOME%\install\external_apps\authplugins\CentralAuthenticationService** to **%DOCUMENTUM%\dba\auth** directory.

CAS Plug-in Trust-Store

CAS Plug-in provides option for anonymous and non-anonymous SSL communication between CAS Plug-in and CAS Server. For non-anonymous SSL, Certificate Authority's (CA) certificate which was used to issue CAS Server's public certificate needs to be stored in a file of filetype pem. Full CA Certificate chain of Issuer should be stored else authentication will fail. To enable this option, *non_anonymous* property should be set to true and *cert_path* property should point to valid pem file with CA certificate chain in CAS configuration file as described in next section.

CAS Plug-in Configuration

Create CAS Plugin configuration file (**dm_cas_auth.ini**) inside **%DOCUMENTUM%\dba\auth** directory and put below properties:

server_host	Server host is the CAS server host name
server_port	HTTP(S) Server port number for CAS server
url_path	url path used in http request sent to CAS server to validate proxy ticket (/‹cas_application_name›/proxyValidate)
service_param	Service name for which the proxy ticket was generated (E.g. ContentServer)
is_https	Specify whether CAS Plug-in will communicate with CAS Server over https or http protocol
non_anonymous	Specify whether non-anonymous or anonymous SSL is used for communication between CAS Plug-in and CAS Server. Only needed when *is_https* is set to true
cert_path	Path to file of filetype *pem* containing CA Certificate chain for CAS Server's public certificate. Only needed if *non_anonymous* is set to true

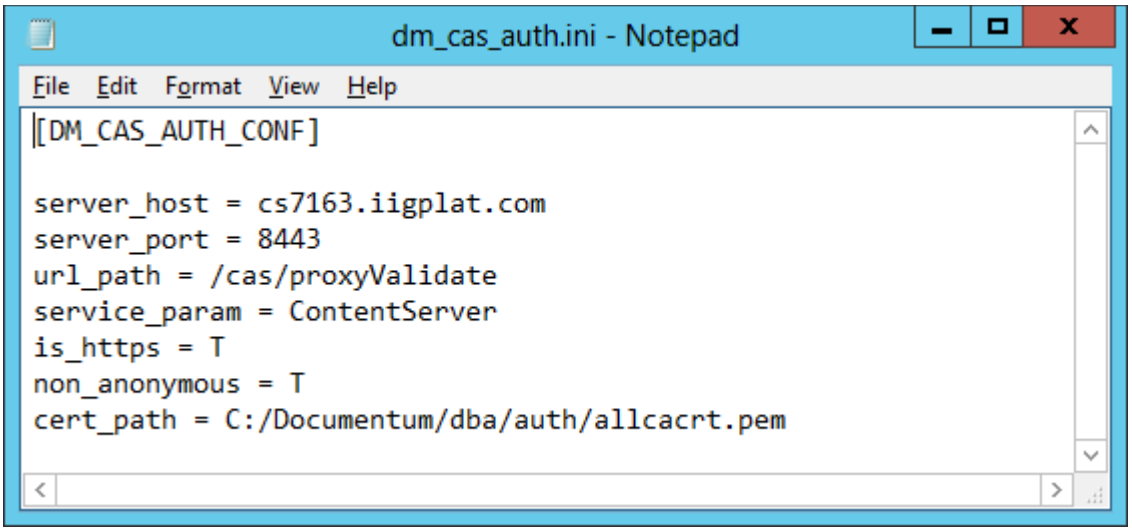

Figure 4. Sample CAS Plug-in configuration file (dm_cas_auth.ini)

Verify Plug-in load

- Start Docbase
- Verify in Docbase logs that CAS plug-in is loaded

```
[DM_SESSION_I_AUTH_PLUGIN_LOADED]info:    "Loaded Authentication Plugin
with code 'dm_cas' (C:\Documentum\dba\auth\dm_cas_auth.dll)."
```

- Verify in CAS plug-in logs that plug-in load was successful (See in Troubleshooting how to enable plugin trace)

```
Documentum Authentication Plugin Trace File
(c) Copyright EMC Corp., 2013
All rights reserved.

11/05/13 01:11:10       Initializing dm_cas plugin
11/05/13 01:11:10       Following are the auth init params
11/05/13 01:11:10       is_https = True
11/05/13 01:11:10       server_host = cs7163.iigplat.com
11/05/13 01:11:10       server_port = 8443
11/05/13 01:11:10       url_path = /cas/proxyValidate
11/05/13 01:11:10       service_param = ContentServer
```

```
11/05/13 01:11:10        cert_path = C:/Documentum/dba/auth/allcacrt.pem
11/05/13 01:11:10        non_anonymous = True
```

LDAP Configuration

Configure Content Server to Sync with same LDAP server as used by CAS server. User login name should map to user LDAP property samAccountName. For more information and detail steps for configuring LDAP server, refer to *EMC Documentum Administrator User Guide*.

Troubleshooting

CAS Plug-in Trace

To enable authentication plug-in trace, follow below steps

- Open Documentum Server Manager
- Stop Repository Service
- In 'Repository' tab, click on 'Edit Service'
- Edit 'Command' and add '-otrace_authentication' option at the end
- Start Repository Service

Log file **dm_cas_<docbase>.log** will be created in directory **$DOCUMENTUM/dba/log**

CAS Server Logs

Location and logging levels for CAS server logs can be defined in below file:

WEB-INF\classes\log4j.xml

SSL related Errors

Error when trying to access Service Management webapp

```
HTTP Status 500 - javax.net.ssl.SSLHandshakeException:
sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to
find valid certification path to requested target
```

Reason: CAS server public certificate missing from CAS server's trust-store.

Resolution: Import CAS server's public certificate to CAS Server's trust-store.

Error during CAS server's callback to Application Server

```
ERROR [org.jasig.cas.util.HttpClient] -
<sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to
find valid certification path to requested target>

javax.net.ssl.SSLHandshakeException:
sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilder

Exception: unable to find valid certification path to requested target
```

Reason: Application server's public certificate missing from CAS server's trust-store.

Resolution: Import application server's public certificate to CAS server's trust-store.

Error during login to CAS Server

```
ERROR [org.jasig.cas.authentication.AuthenticationManagerImpl] -
<org.jasig.cas.adaptors.ldap.BindLdapAuthenticationHandler threw error
authenticating [username: admin]>

org.springframework.ldap.CommunicationException:
domainctlr.iigplat.com:636; nested exception is
javax.naming.CommunicationException: domainctlr.iigplat.com:636 [Root
exception is javax.net.ssl.SSLHandshakeException:
sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to
find valid certification path to requested target]
```

Reason: LDAP server's public certificate missing from CAS server's trust-store.

Resolution: Import LDAP server's public certificate to CAS Server's trust-store.

CAS Server Errors

CAS Server responds with Invalid Ticket message

Error in CAS Plug-in trace:

```
Respose code : 200

Respose content :

<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>

    <cas:authenticationFailure code='INVALID_TICKET'>

          ticket &#039;ST-9-xKWPvnCX9VuUh45J6msn-
          cs7163.iigplat.com&#039; not recognized
```

```
    </cas:authenticationFailure>
</cas:serviceResponse>
4336 11/06/13 16:34:53        Validate Cas user Failed
4336 11/06/13 16:34:53        Error : CAS Authentication failed with
following Error Code : 'INVALID_TICKET' Error String :
ticket &#039;ST-9-xKWPvnCX9VuUh45J6msn-cs7163.iigplat.com&#039; not
recognized
```

Reason: One of the common reasons for invalid ticket error is ticket expires by the time it reached CAS server. To verify that this is the reason for above error, check CAS server logs for below message.

```
INFO [org.jasig.cas.CentralAuthenticationServiceImpl] - <ServiceTicket
[ST-9-xKWPvnCX9VuUh45J6msn-cs7163.iigplat.com] has expired.>
```

Resolution: Increase the service ticket timeout by changing value of property st.timeToKillInSeconds in WEB-INF/cas.properties. Refer to Ticket Expiration Policy section (https://wiki.jasig.org/display/CASUM/Ticket+Expiration+Policy) in CAS user manual for more details.

CAS Plug-in Errors

Connection to CAS Server fails in non-anonymous SSL mode

Error in CAS Plug-in trace with errorcode 77:

```
11/07/13 15:07:10      Sending curl request to following url
https://cs7163.iigplat.com:8443/cas/proxyValidate?service=ContentServer&
ticket=ST-15-99SqGeyda67OrggsyuUe-cs7163.iigplat.com
11/07/13 15:07:10       Error : Failed to perform URL:
https://cs7163.iigplat.com:8443/cas/proxyValidate?service=ContentServer&
ticket=ST-15-99SqGeyda67OrggsyuUe-cs7163.iigplat.com, errorcode: 77:
Problem with the SSL CA cert (path? access rights?)
```

Reason: Trust-store (pem file with CA certificate chain) file is missing or incorrect path specified in plug-in configuration file.

Resolution: Check whether trust-store file exists in location specified in plug-in configuration file (dm_cas_auth.ini).

Error in CAS Plug-in trace with errorcode 60:

```
11/07/13 15:10:40      Sending curl request to following url
https://cs7163.iigplat.com:8443/cas/proxyValidate?service=ContentServer&
ticket=ST-17-IdGpXpjFCrVYaV6Xsyjs-cs7163.iigplat.com

11/07/13 15:10:40       Error : Failed to perform URL:
https://cs7163.iigplat.com:8443/cas/proxyValidate?service=ContentServer&
ticket=ST-17-IdGpXpjFCrVYaV6Xsyjs-cs7163.iigplat.com, errorcode: 60:
Peer certificate cannot be authenticated with given CA certificates
```

Reason: When CAS Plug-in is configured in non-anonymous SSL mode, if CAS Plug-in is not able to verify public certificate sent by CAS Server using CA certificates stored in pem file, then CAS plug-in rejects the connection and error is displayed. Some of the reasons for this are:

- Complete CA certificate chain not included in trust-store (.pem) file
- CAS server's public certificate or Issuer's certificate expired
- CAS server's certificate not yet valid

Resolution: To check whether proper CA certificates are stored in pem file, execute below OpenSSL command, where allcacrt.pem is the trust-store file configured in CAS Plug-in configuration and cas.pem contains public certificate of CAS server.

```
openssl verify -CAfile allcacrt.pem cascrt.pem
```

If this command displays error, then proper CA certificates needs to be stored in allcacrt.pem.

Plug-in load fails

Error in docbase logs (%DOCUMENTUM%\dba\log\‹docbase›.log):

```
[DM_SESSION_E_AUTH_PLUGIN_LOAD_INIT_ERROR]error:  "Failed to load
Authentication Plugin C:\Documentum\dba\auth\dm_cas_auth.dll. Plugin
initialization returned error: ''."
```

Error in CAS plug-in logs (%DOCUMENTUM%\dba\log\dm_cas_‹docbase›.log):

```
Initialization failed: Failed to open file
C:\Documentum\dba\auth\dm_cas_auth.ini.Plugin initialization failed.
```

Reason: Plugin configuration is missing.

Resolution: Create Plug-in configuration file as described Content Server configuration section.

Conclusion

This paper explains CAS architecture, protocol and how CAS SSO can be enabled with Documentum applications. Only the basic configuration required for making CAS server work with Documentum is shown here. For further information on CAS protocol & configuration, sources provided in next section can be referenced.

References

- CAS project site: http://www.jasig.org/cas
- CAS User Manual Wiki: https://wiki.jasig.org/display/CASUM/Home
- OpenSSL Documentation: http://www.openssl.org/docs/apps/openssl.html